**Jaypee University of Engineering and Technology**

**B. Tech. (CSE) - II Semester**

**Object Oriented Programming (18B11CI211)**

**Tutorial – 4(Function Overloading)**

|  |  |  |
| --- | --- | --- |
| Q1. | How to create a dynamic array of pointers (to integers) of size 10 using new in C++?  Hint: We can create a non-dynamic array using int \*arr[10] **(A)** int \*arr = new int \*[10]; **(B)** int \*\*arr = new int \*[10]; **(C)** int \*arr = new int [10]; **(D)** Not Possible | |
| Q.2 | |  |  | | --- | --- | | Output?  #include<iostream>  using namespace std;    int fun(int x = 0, int y = 0, int z)  {  return (x + y + z); } | int main()  {     cout << fun(10);     return 0;  } |   **(A)** 10 (**B)** 0 **(C)** 20 **(D)** Compiler Error | |
| Q.3 | Which of the following overloaded functions are NOT allowed in C++?  1) Function declarations that differ only in the return type  int fun(int x, int y);  void fun(int x, int y);  2) Functions that differ only by static keyword in return type  int fun(int x, int y);  static int fun(int x, int y);  3)Parameter declarations that differ only in a pointer \* versus an array []  int fun(int \*ptr, int n);  int fun(int ptr[], int n);  4) Two parameter declarations that differ only in their default arguments  int fun( int x, int y);  int fun( int x, int y = 10);  **(A)** All of the above **(B)** All except 2) **(C)** All except 1) **(D)** All except 2 and 4 | |
| Q.4 | What does the inline keyword do?  **(A)** Indicates a function declaration  **(B)** Tells the compiler to use the function only within the same source code file  **(C)** Causes all function calls to be replaced by the code from the function **(D)**Allows one-line function declarations |  | |
|  | **Q.5** Why would you want to use inline functions?  (A) To decrease the size of the resulting program (B) To increase the speed of the resulting program (C) To simplify the source code file (D)To remove unnecessary functions  **Q6.** Create an Account class that a bank might use to represent customers' bank accounts. Include a data member of type *int* to represent the account balance. Provide a constructor that receives an initial balance and uses it to initialize the data member. The constructor should validate the initial balance to ensure that it's greater than or equal to 0. If not, set the balance to 0 and display an error message indicating that the initial balance was invalid. Provide three member functions. Member function *credit* should add an amount to the current balance. Member function *debit* should withdraw money from the Account and ensure that the debit amount does not exceed the Account's balance. If it does, the balance should be left unchanged and the function should print a message indicating "Debit amount exceeded account balance." Member function *getbalance* should return the current balance. Create a program that creates two Account objects and tests the member functions of class Account.  **Q7.**What is the output of this program?  #include<iostream>  using namespace std;  class Car  {  public:  int speed;  };  int main()  {  int Car:: \*pSpeed=&Car:: speed;  Car c1;  c1.speed=1;  cout<<c1.speed<<endl;  c1.\*pSpeed=2;  cout<<c1.speed<<endl;  return 0;  }  **a)1 b)2 c)Both 1 and 2 d)None of the mentioned** | |  |
|  |  | |